Distance Vector Algorithms

Pros

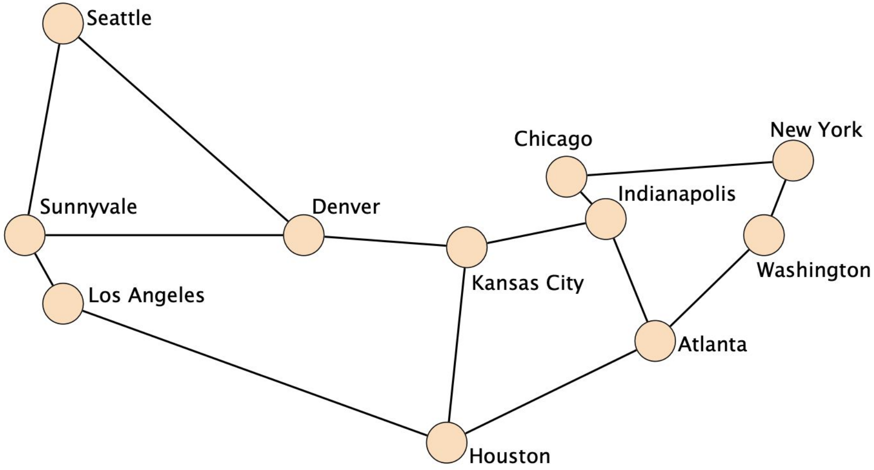
- Simple to configure / maintain
- Only need a local view of the world

Cons

- Slow to converge
- Loops are possible
- Count to infinity
- Wastes bandwidth - constant updates even when nothing changes

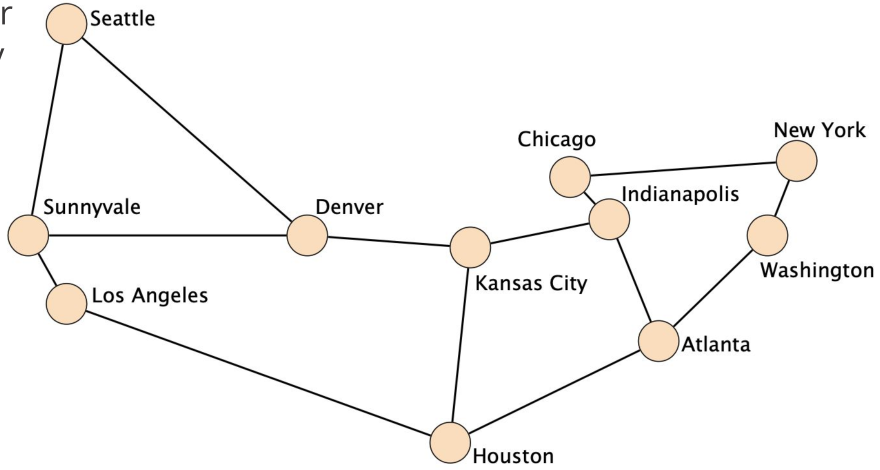
Link Weight Configurations

- The Abilene network was a high-performance backbone network in the US. You are the network operator in charge and you have to configure the link weights in the network. Initially, all links have a weight of one and routers will always use the shortest-path available to reach a destination.



Link Weight Configurations

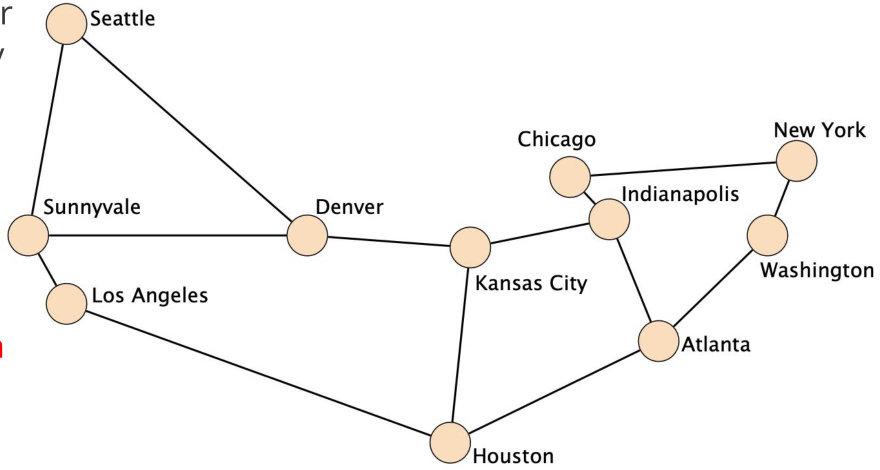
Is it possible to configure the link weights such that the packets sent by the router located in Los Angeles to the router located in New York follow one path while the packets sent by the router located in New York to the router located in Los Angeles follow a completely different path?



Link Weight Configurations

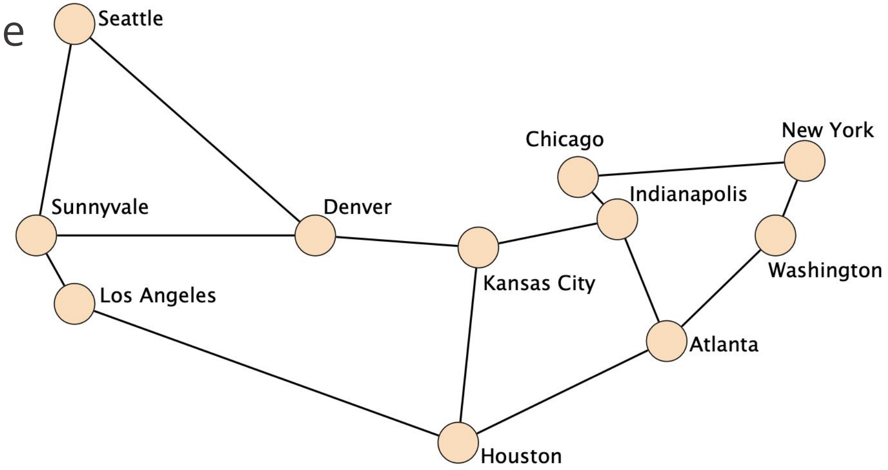
Is it possible to configure the link weights such that the packets sent by the router located in Los Angeles to the router located in New York follow one path while the packets sent by the router located in New York to the router located in Los Angeles follow a completely different path?

Solution: Not possible. We consider only links which have the same weight in both directions. If the two routers would use different paths for the two traffic directions, the two paths would need different total weights. That implies that one path is shorter and one router is not using the shortest-path available. A contradiction to our initial assumption.



Link Weight Configurations

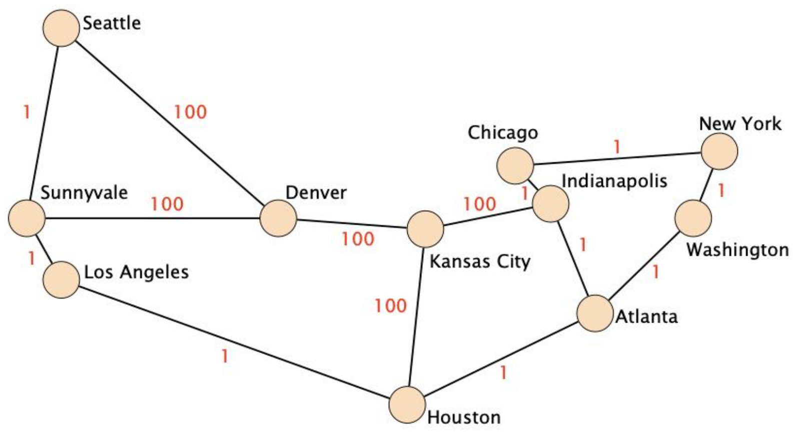
Assume that the routers located in Denver and Kansas City need to exchange lots of data on the direct link. Can you configure the link weights such that the link between these two routers does not carry any packet sent by any other router in the network?



Link Weight Configurations

Assume that the routers located in Denver and Kansas City need to exchange lots of data on th direct link. Can you configure the link weights such that the link between these two routers does not carry any packet sent by any other router in the network?

Solution:



Routing

Routing Comes in Two Flavors: *intra* and *inter*-domain routing

inter-domain routing	intra-domain routing
Find paths between networks	Find paths within a network

Routing Comes in Two Flavors: *intra* and *inter*-domain routing

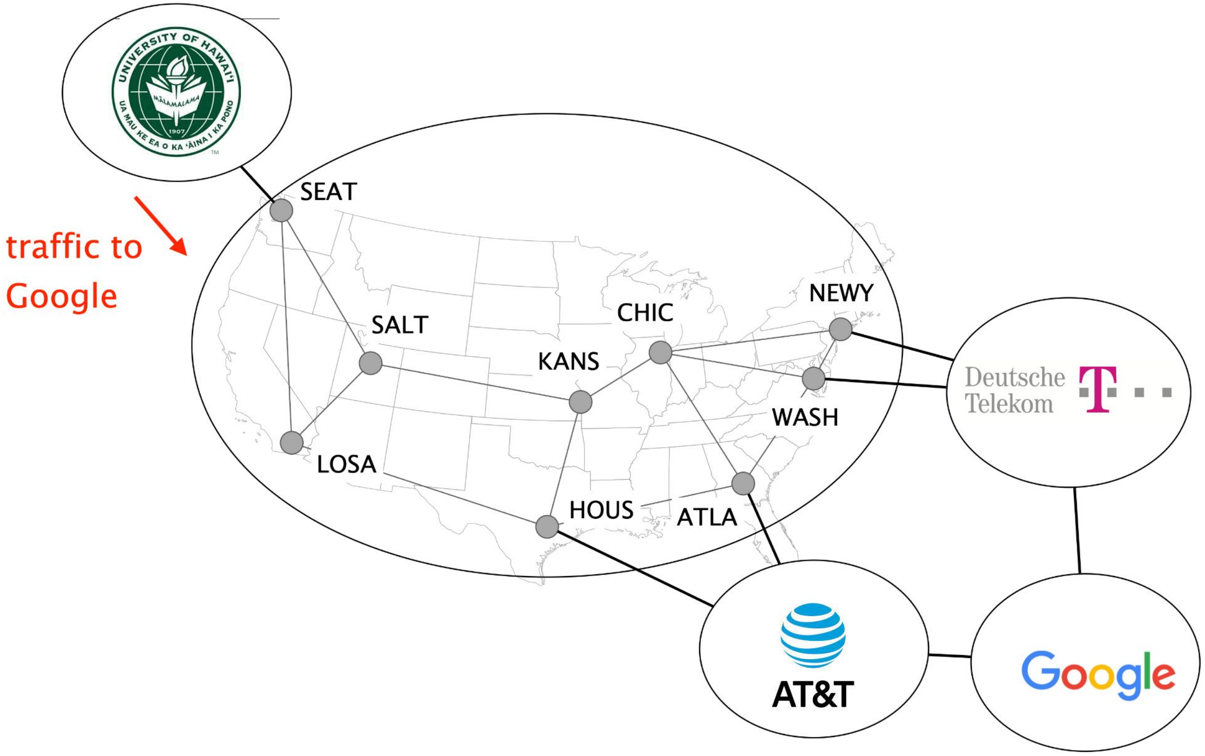
inter-domain routing

Find paths between networks

intra-domain routing

Find paths within a network

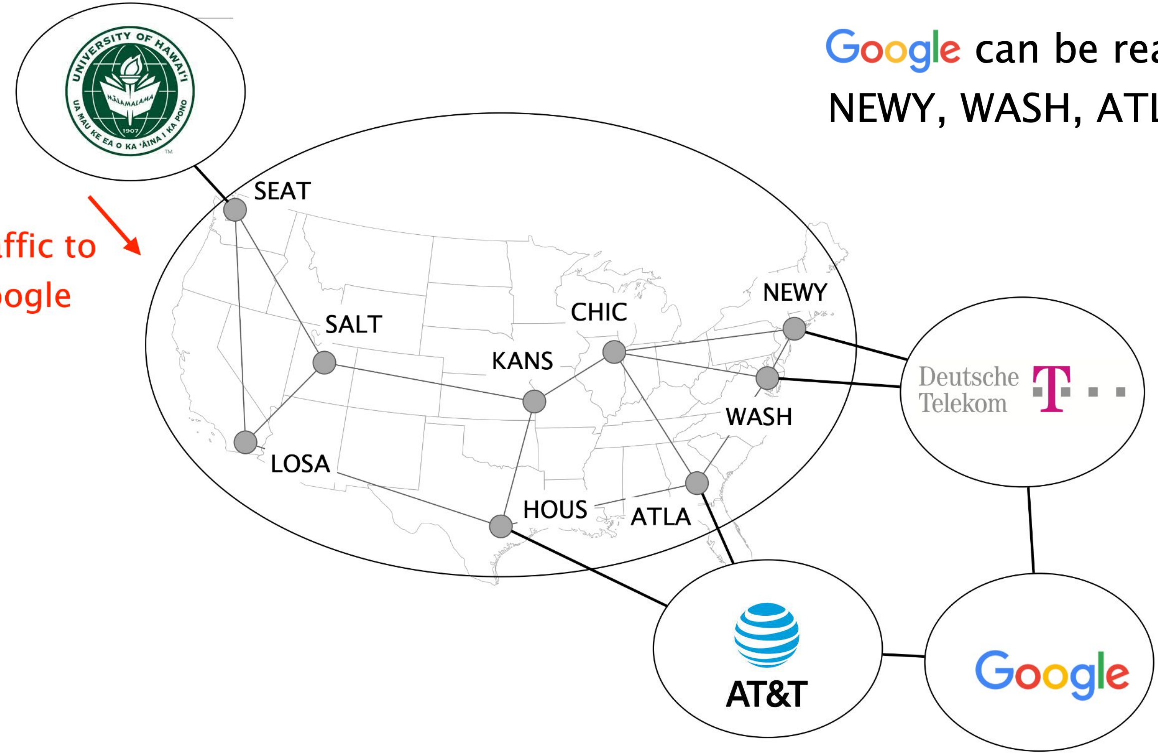
Routing Comes in Two Flavors: *intra* and *inter*-domain routing



Routing Comes in Two Flavors: *intra* and *inter*-domain routing

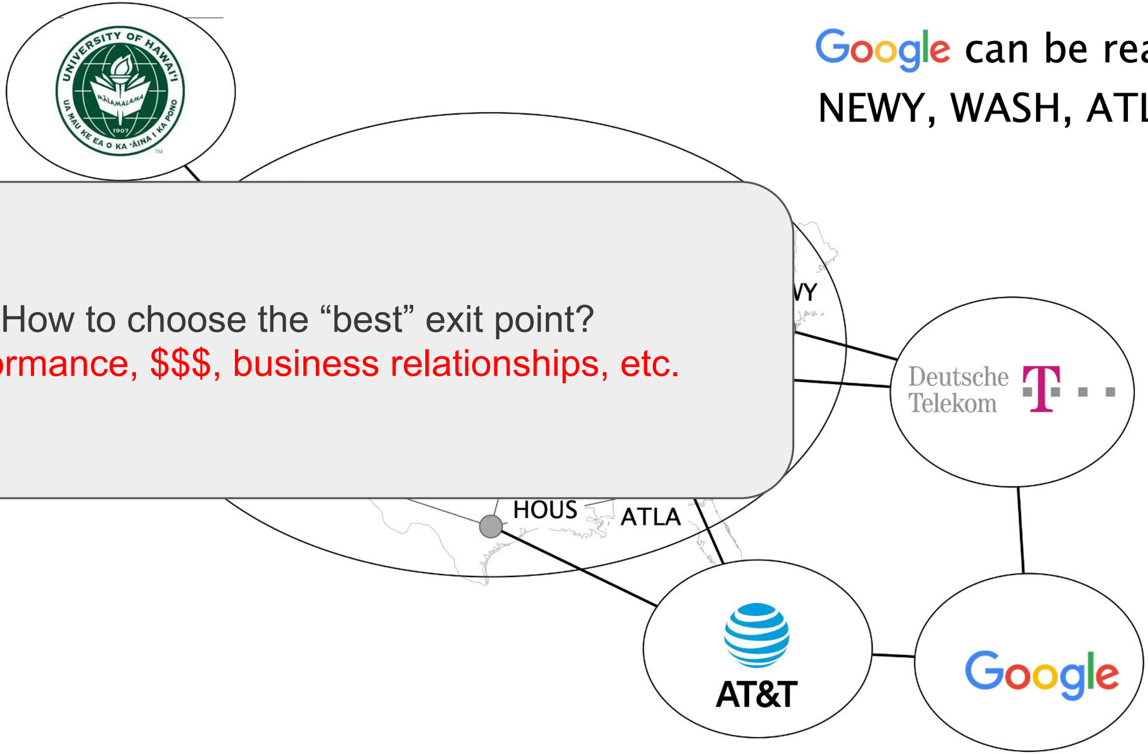
Google can be reached via NEWY, WASH, ATLA, HOUS

traffic to Google



Routing Comes in Two Flavors: *intra* and *inter*-domain routing

Google can be reached via
NEWY, WASH, ATLA, HOUS



How to choose the "best" exit point?
Performance, $$$, business relationships, etc.

Routing Comes in Two Flavors: *intra* and *inter*-domain routing

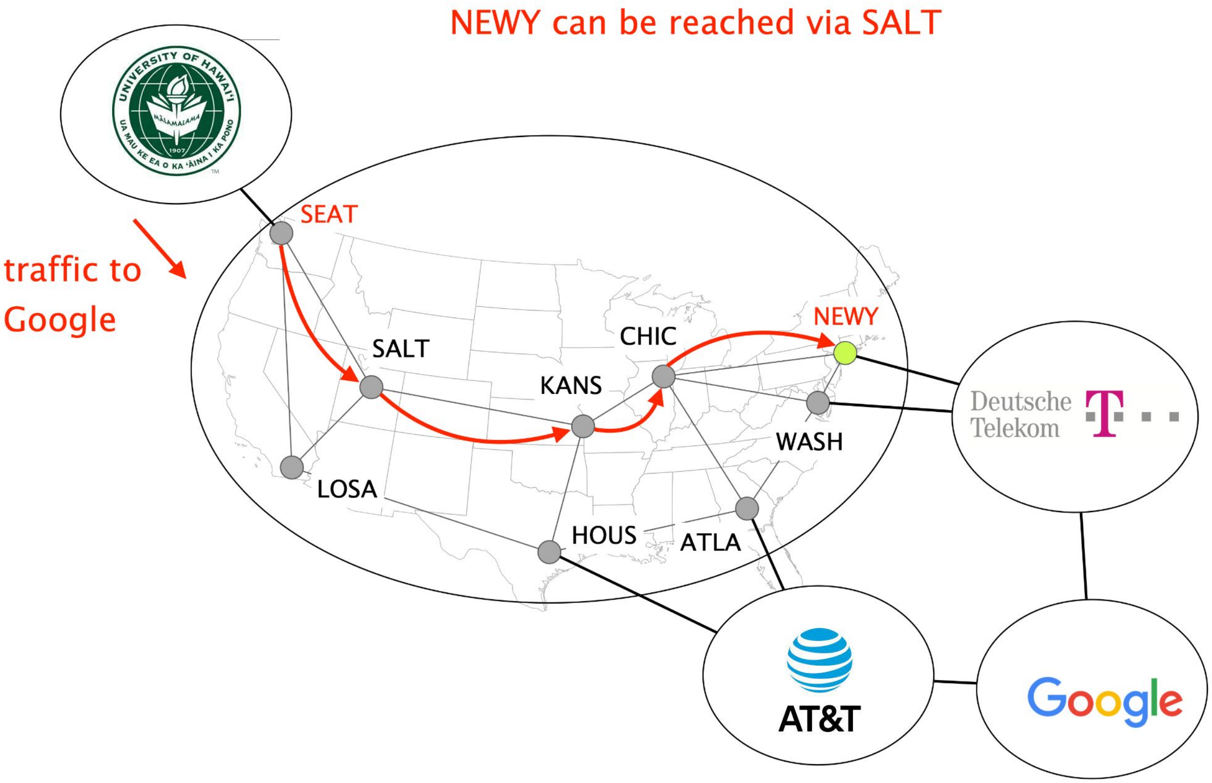
inter-domain routing

intra-domain routing

Find paths between networks

Find paths within a network

Routing Comes in Two Flavors: *intra* and *inter*-domain routing



```
traceroute to orange.fr (193.252.133.20), 64 hops max, 52 byte packets
```

Intra-domain routing ↓	168.105.224.2 (168.105.224.2) 4.483 ms 3.005 ms 4.315 ms vl-3223-manoa7050-2.uhnet.net (128.171.186.190) 3.182 ms vl-3222-manoa7050-1.uhnet.net (128.171.186.188) 3.523 ms 2.602 ms
Intra-domain routing ↓	xe-0-0-0-667-coconut-re0.uhnet.net (128.171.64.182) 8.254 ms xe-1-0-0-669-coconut-re0.uhnet.net (128.171.213.13) 5.655 ms 5.989 ms xe-0-0-6-73-ohelo-re0.uhnet.net (205.166.205.46) 5.414 ms 4.974 ms 6.153 ms
Intra-domain routing ↓	dc-svl-agg4--uh-10ge.cenic.net (137.164.50.234) 52.903 ms 52.820 ms 57.583 ms dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80) 53.511 ms 53.705 ms 53.249 ms
Intra-domain routing ↓	ae76-91.edge9.sanjose1.level3.net (4.15.122.45) 55.922 ms 69.620 ms 56.789 ms orange-level3-sanjose1.level3.net (4.68.68.10) 52.534 ms 52.444 ms 53.727 ms

```
traceroute to orange.fr (193.252.133.20), 64 hops max, 52 byte packets
168.105.224.2 (168.105.224.2)  4.483 ms  3.005 ms  4.315 ms
vl-3223-manoa7050-2.uhnet.net (128.171.186.190)  3.182 ms
vl-3222-manoa7050-1.uhnet.net (128.171.186.188)  3.523 ms  2.602 ms
xe-0-0-0-667-coconut-re0.uhnet.net (128.171.64.182)  8.254 ms
xe-1-0-0-669-coconut-re0.uhnet.net (128.171.213.13)  5.655 ms  5.989 ms
xe-0-0-6-73-ohelo-re0.uhnet.net (205.166.205.46)  5.414 ms  4.974 ms  6.153 ms
dc-svl-agg4--uh-10ge.cenic.net (137.164.50.234)  52.903 ms  52.820 ms  57.583 ms
dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80)  53.511 ms  53.705 ms  53.249 ms
ae76-91.edge9.sanjose1.level3.net (4.15.122.45)  55.922 ms  69.620 ms  56.789 ms
orange-level3-sanjose1.level3.net (4.68.68.10)  52.534 ms  52.444 ms  53.727 ms
```

Intra-domain routing (hops 1–3)

Intra-domain routing (hops 4–6)

Intra-domain routing (hops 7–8)

Intra-domain routing (hops 9–10)

Inter-domain routing (between hops 3–4, 6–7, 8–9)

Internet Routing

1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols

Internet Routing

1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols

Intra-domain routing enables routers to compute forwarding paths to any internal subnet

Internet Routing

1. Intra-domain routing
    - Link-state protocols
    - Distance-vector protocols

2. Inter-domain routing
    - Path-vector protocols

Intra-domain routing enables routers to compute forwarding paths to any internal subnet

What kind of paths?

Network Operators don't use Arbitrary Paths, they use Good Paths

definition

A good path is a path that

minimizes some network-wide metric

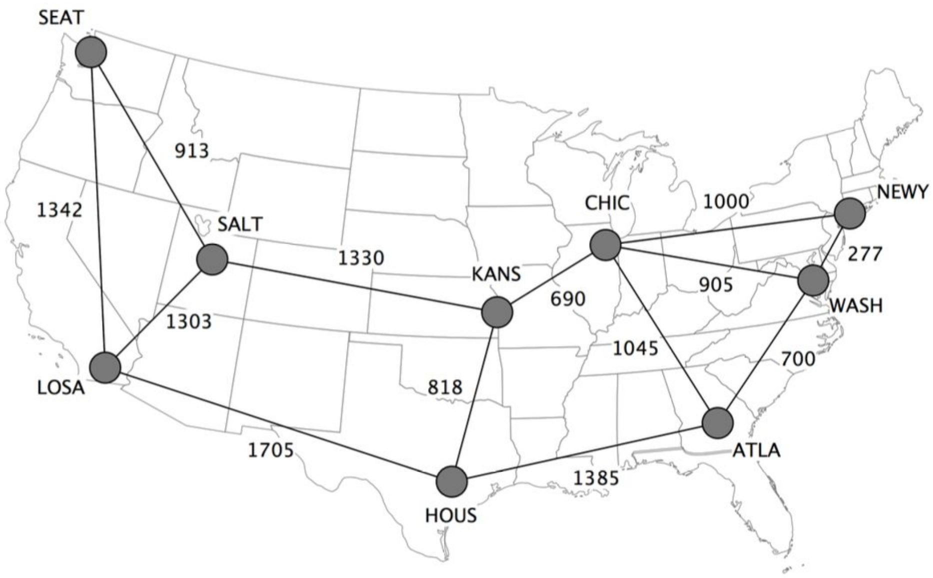
typically delay, load, loss, cost

approach

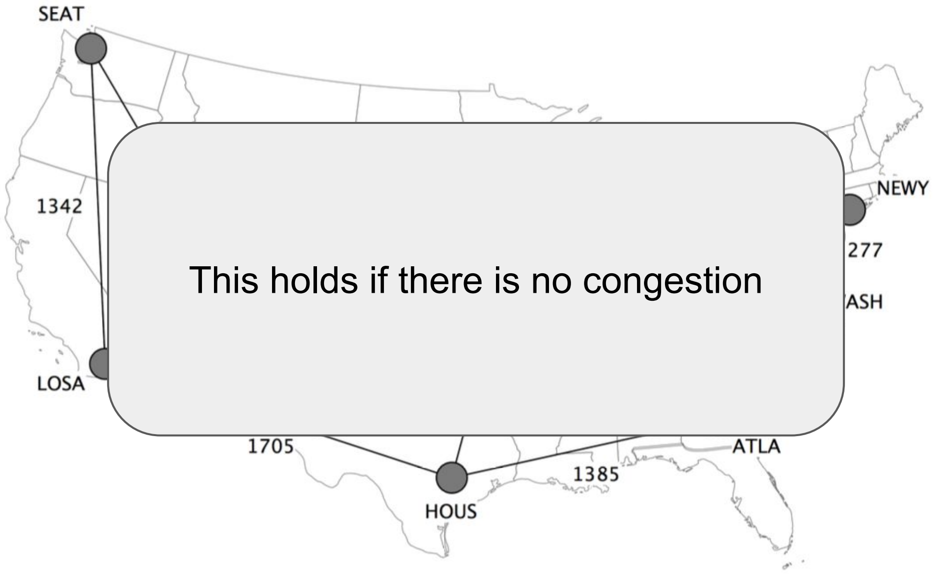
Assign to each link a weight (usually static),

compute the *shortest-path* to each destination

When Link Weights are Proportional to Distance, Shortest Paths Minimize end-to-end Delay

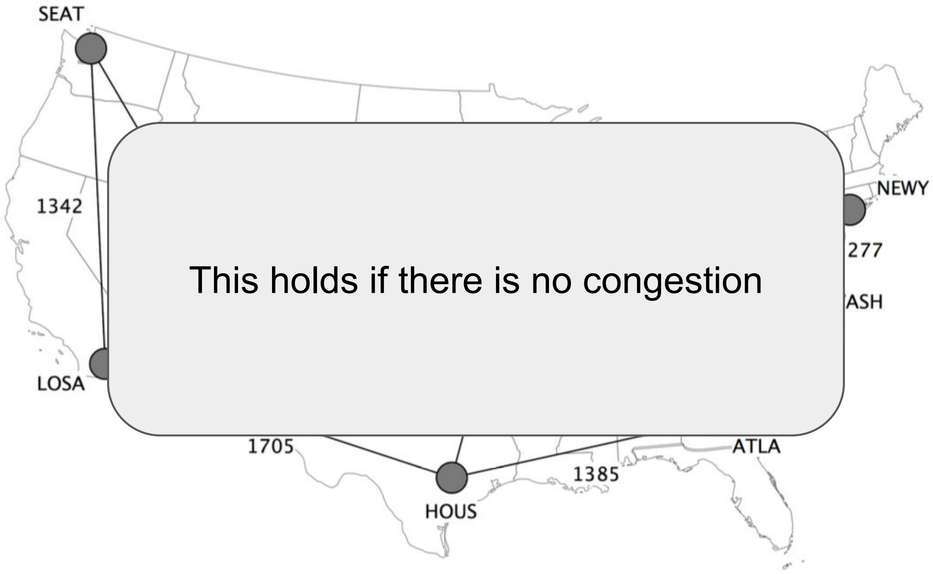


When Link Weights are Proportional to Distance, Shortest Paths Minimize end-to-end Delay



This holds if there is no congestion

When Link Weights are Inversely Proportional to Link Capacity, Throughput is Maximized



This holds if there is no congestion

Link-State Routing...

Each router keeps track of its incident links and cost as well as whether it is up or down

Each router broadcast its own links state to give every router a complete view of the graph

Routers run Dijkstra on the corresponding graph to compute their shortest-paths and forwarding tables

Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

All nodes are ensured to receive the latest version of all link-states

Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

All nodes are ensured to receive the latest version of all link-states

- challenges
    - packet loss
    - out of order arrival

Link-State Routing - Flooding

Node sends its link-state on all its links

Next node does the same, except on the link where the information arrived

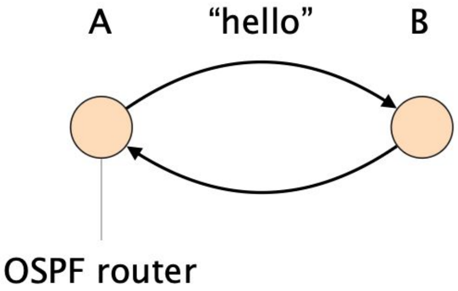
All nodes are ensured to receive the latest version of all link-states

- challenges
  - packet loss
  - out of order arrival
- solutions
  - ACK & retransmissions
  - sequence number
  - time-to-live for each link-state

When to Initiate Flooding?

Topology change	link or node failure/recovery
Configuration change	link cost change
Periodically	refresh the link-state information every (say) 30 minutes account for possible data corruption

How do actual Link-State Protocols Detect Topology Changes? Software-based Beaconing

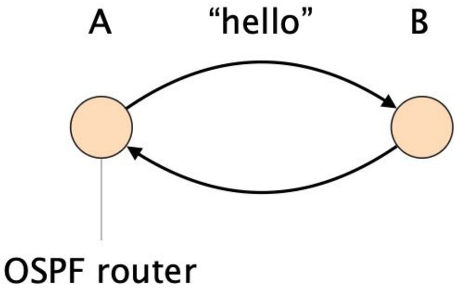


Routers periodically exchange “Hello” in both directions (*e.g.* every 30s)

Trigger a failure after few missed “Hellos” (*e.g.*, after 3 missed ones)

What kind of tradeoffs are present here?

How do actual Link-State Protocols Detect Topology Changes? Software-based Beaconing



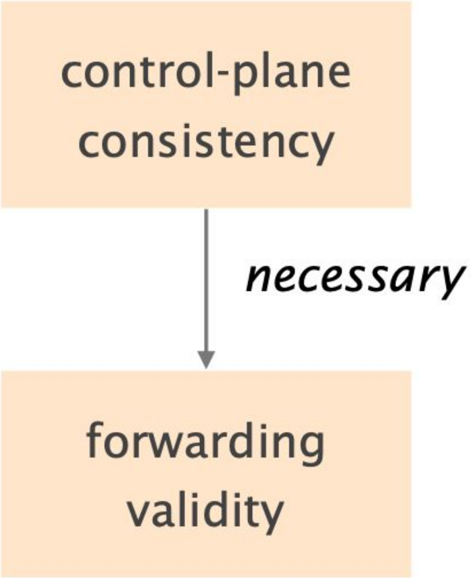
Routers periodically exchange "Hello" in both directions (*e.g.* every 30s)

Trigger a failure after few missed "Hellos" (*e.g.*, after 3 missed ones)

Tradeoffs between:

- detection speed
- bandwidth and CPU overhead
- false positive/negatives

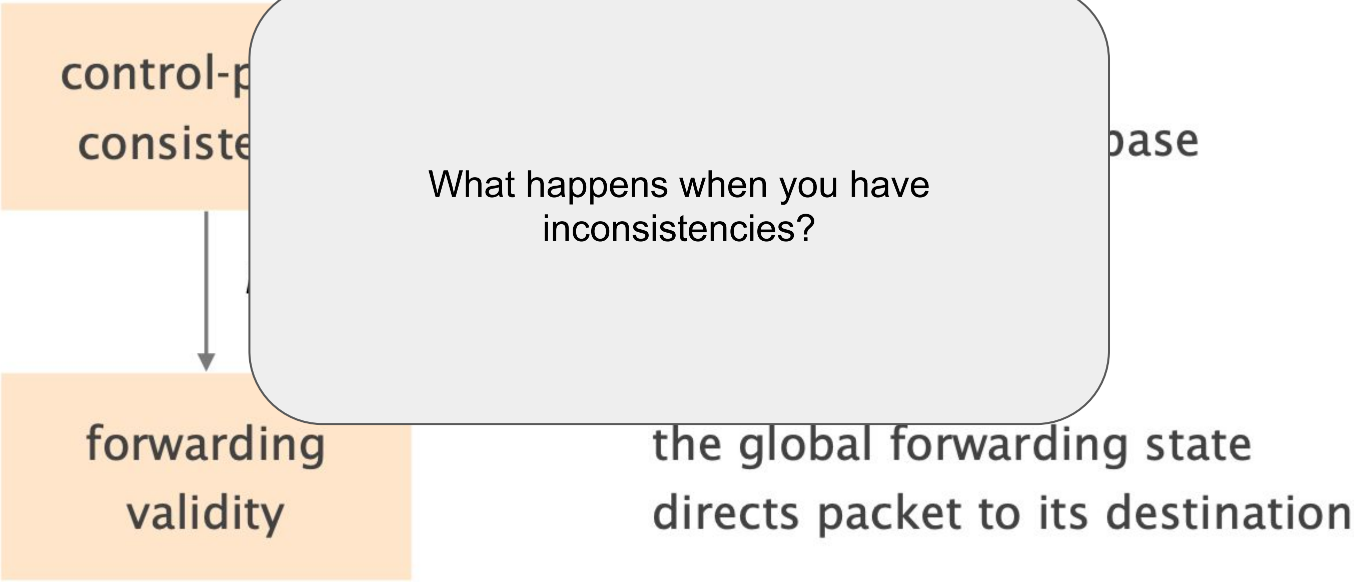
During Network Changes the Link-State DBs of Each Router May Differ



all nodes have the same link-state database

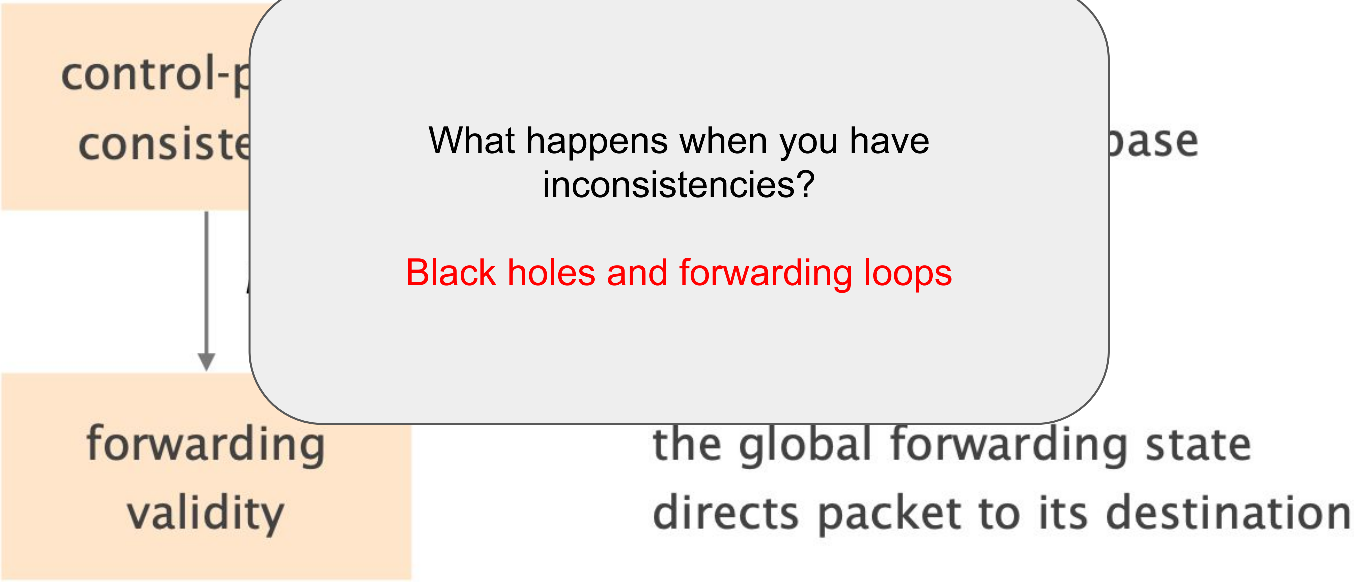
the global forwarding state directs packet to its destination

During Network Changes the Link-State DBs of Each Router May Differ



What happens when you have inconsistencies?

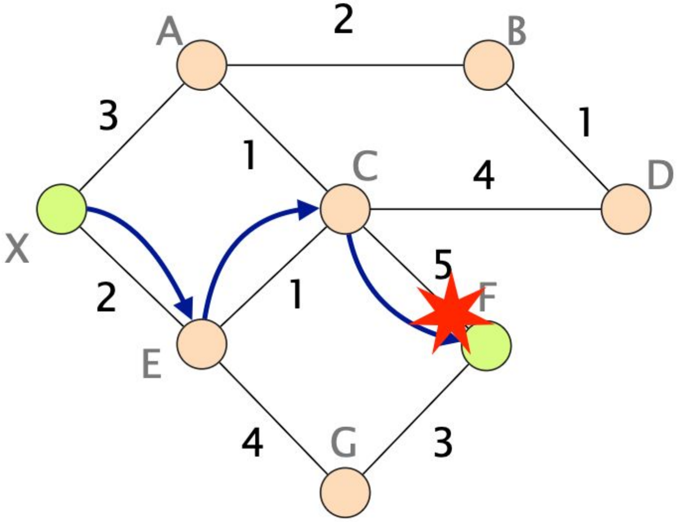
During Network Changes the Link-State DBs of Each Router May Differ



What happens when you have inconsistencies?

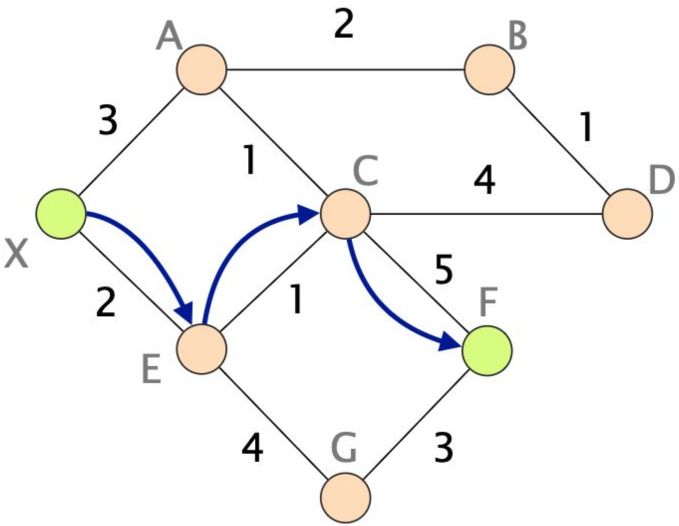
Black holes and forwarding loops

Black Holes - Due to Detection Delay as Routers Do Not Immediately Detect Failure



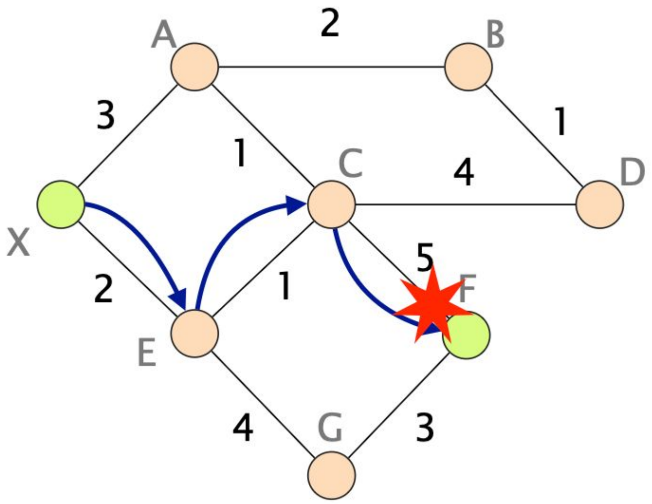
depends on the timeout for detecting lost hellos

Forwarding Loops - Due to Inconsistent Link-State DBs



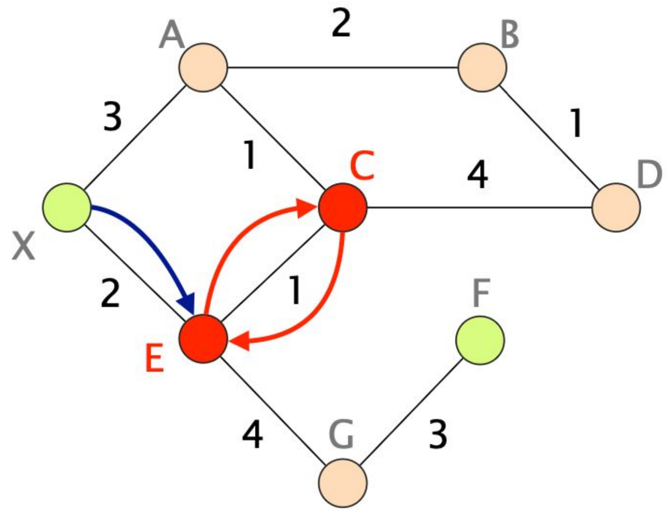
Initial forwarding state

Forwarding Loops - Due to Inconsistent Link-State DBs



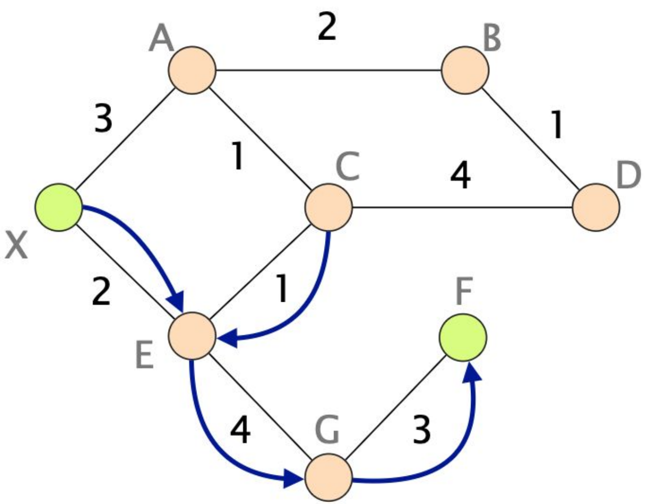
C learns about the failure
and immediately reroute to E

Forwarding Loops - Due to Inconsistent Link-State DBs



A loop appears as E
isn't yet aware of the failure

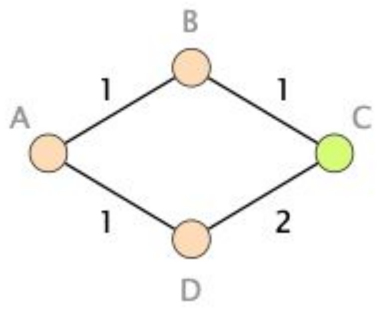
Forwarding Loops - Due to Inconsistent Link-State DBs



The loop disappears as soon as
E updates its forwarding table

Forwarding Loops - Due to Inconsistent Link-State DBs

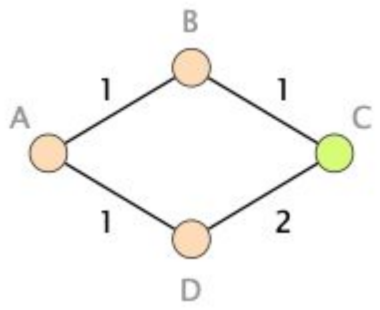
Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.



Assume that routers A, B and D transit traffic for an IP destination connected to C and that link (B,C) fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?

Forwarding Loops - Due to Inconsistent Link-State DBs

Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

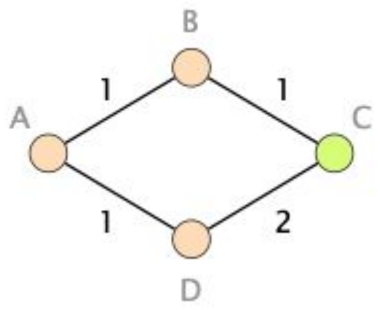


Assume that routers A, B and D transit traffic for an IP destination connected to C and that link (B,C) fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?

Solution: Nodes A and B could see their packets stuck in a forwarding loop if B updates its forwarding table before A, which is likely to happen as B would be the first to learn about an adjacent link failure. On the other hand, D would not see any loop as it uses its direct link with C to reach any destination connected beyond it.

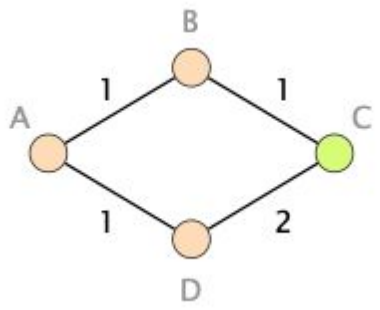
Forwarding Loops - Due to Inconsistent Link-State DBs

Assume now that the network administrator wants to take down the link (B,C), on purpose, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link before taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link (B,C) that would ensure that no packet destined to C is dropped?



Forwarding Loops - Due to Inconsistent Link-State DBs

Assume now that the network administrator wants to take down the link (B,C), on purpose, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link before taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link (B,C) that would ensure that no packet destined to C is dropped?



Solution: One example of a minimum sequence of (B,C) weights is [1, 3, 5].

Note: The problem highlighted above happens because B shifts traffic to A before A shifts traffic to D, hence creating a forwarding loop. By setting the (B,C) link weight to 3 first, (only) A shifts from using (A, B, C) to using (A, D, C). Once A has shifted, it is safe to shift B by setting the link weight to 5 (or higher). Once B has shifted as well, the link can be safely torn down.